Classic Classifier

In [ ]:

```
%env CUDA_VISIBLE_DEVICES=1
```

```
env: CUDA_VISIBLE_DEVICES=1
```

In [ ]:

```
import numpy as np
import cudf
import cuml
from cupy import asnumpy
from joblib import dump, load
```

In [ ]:

```
def calc_f1(cm):
    # Extracting True Positives, False Positives, False Negatives
    TP = cm[0][0]
    FP = cm[0][1]
    FN = cm[1][0]
    # TN = confusion_matrix[1][1] # True Negatives are not needed for F1

    # Calculating Precision and Recall
    precision = TP / (TP + FP) if (TP + FP) > 0 else 0
    recall = TP / (TP + FN) if (TP + FN) > 0 else 0

    # Calculating F1 Score
    if precision + recall == 0: # Avoiding division by zero
        f1_score = 0
    else:
        f1_score = 2 * (precision * recall) / (precision + recall)

    return f1_score
```

Load Data

In [ ]:

```
train_data_path = "./data/train.csv"
```

In [ ]:

```
df = cudf.read_csv(train_data_path)
df.describe()
```

```
/home/dx/miniconda3/envs/rapids-24.02/lib/python3.10/site-packages/cudf/co
re/dataframe.py:5106: FutureWarning: `datetime_is_numeric` is deprecated.
Specify `datetime_is_numeric=True` to silence this warning and adopt the f
uture behavior now.
  warnings.warn(
/home/dx/miniconda3/envs/rapids-24.02/lib/python3.10/site-packages/cudf/co
re/series.py:3319: FutureWarning: `datetime_is_numeric` is deprecated and
will be removed in a future release. Specify `datetime_is_numeric=True` to
silence this warning and adopt the future behavior now.
  warnings.warn(
```

Out[ ]:

	ind_recommended	activation	customer_digital_activity_04	customer_s
count	1.222998e+07	1.222998e+07	1.472619e+06	1.018
mean	1.264980e-01	5.725000e-03	9.745803e+00	1.342
std	3.324100e-01	7.544700e-02	4.156839e+01	6.454
min	0.000000e+00	0.000000e+00	1.000000e+00	1.000
25%	0.000000e+00	0.000000e+00	1.000000e+00	3.844
50%	0.000000e+00	0.000000e+00	2.000000e+00	6.437
75%	0.000000e+00	0.000000e+00	6.000000e+00	1.138
max	1.000000e+00	1.000000e+00	8.560000e+02	2.489

8 rows × 71 columns

Data Cleanup

In [ ]:

```
df.drop(["customer", "merchant"], axis=1, inplace=True)
df.describe()
```

```
/home/dx/miniconda3/envs/rapids-24.02/lib/python3.10/site-packages/cudf/co
re/dataframe.py:5106: FutureWarning: `datetime_is_numeric` is deprecated.
Specify `datetime_is_numeric=True` to silence this warning and adopt the f
uture behavior now.
  warnings.warn(
/home/dx/miniconda3/envs/rapids-24.02/lib/python3.10/site-packages/cudf/co
re/series.py:3319: FutureWarning: `datetime_is_numeric` is deprecated and
will be removed in a future release. Specify `datetime_is_numeric=True` to
silence this warning and adopt the future behavior now.
  warnings.warn(
```

Out[ ]:

	ind_recommended	activation	customer_digital_activity_04	customer_s
count	1.222998e+07	1.222998e+07	1.472619e+06	1.018
mean	1.264980e-01	5.725000e-03	9.745803e+00	1.342
std	3.324100e-01	7.544700e-02	4.156839e+01	6.454
min	0.000000e+00	0.000000e+00	1.000000e+00	1.000
25%	0.000000e+00	0.000000e+00	1.000000e+00	3.844
50%	0.000000e+00	0.000000e+00	2.000000e+00	6.437
75%	0.000000e+00	0.000000e+00	6.000000e+00	1.138
max	1.000000e+00	1.000000e+00	8.560000e+02	2.489

8 rows × 69 columns

In [ ]:

```
print(df.shape)
```

```
(12229978, 69)
```

In [ ]:

```
threshold = len(df.columns) * 0.8
df_cleaned = df.dropna(thresh=threshold)
df_cleaned.shape
```

Out[ ]:

```
(1373560, 69)
```

In [ ]:

```
df_cleaned.describe()
```

```
/home/dx/miniconda3/envs/rapids-24.02/lib/python3.10/site-packages/cudf/co
re/dataframe.py:5106: FutureWarning: `datetime_is_numeric` is deprecated.
Specify `datetime_is_numeric=True` to silence this warning and adopt the f
uture behavior now.
  warnings.warn(
/home/dx/miniconda3/envs/rapids-24.02/lib/python3.10/site-packages/cudf/co
re/series.py:3319: FutureWarning: `datetime_is_numeric` is deprecated and
will be removed in a future release. Specify `datetime_is_numeric=True` to
silence this warning and adopt the future behavior now.
  warnings.warn(
```

Out[ ]:

	ind_recommended	activation	customer_digital_activity_04	customer_s
count	1.373560e+06	1.373560e+06	1.373560e+06	1.3735
mean	2.369470e-01	1.963200e-02	3.283215e+00	1.251
std	4.252100e-01	1.387330e-01	2.212771e+01	4.676
min	0.000000e+00	0.000000e+00	1.000000e+00	1.000
25%	0.000000e+00	0.000000e+00	1.000000e+00	4.373
50%	0.000000e+00	0.000000e+00	1.000000e+00	6.745
75%	0.000000e+00	0.000000e+00	1.000000e+00	1.129
max	1.000000e+00	1.000000e+00	8.470000e+02	1.500

8 rows × 69 columns

In [ ]:

```
are_any_nulls = df_cleaned.isnull().any().any()
are_any_nulls
```

Out[ ]:

```
True
```

In [ ]:

```
for column in df.columns:
    most_frequent = df_cleaned[column].value_counts().index[0]  # Get the
    df_cleaned[column] = df_cleaned[column].fillna(most_frequent)  # Fill
```

In [ ]:

```
are_any_nulls = df_cleaned.isnull().any().any()
are_any_nulls
```

Out[ ]:

```
False
```

In [ ]:

```
from cuml.model_selection import train_test_split

X = df_cleaned.drop(["activation", "ind_recommended"], axis=1)
y = df_cleaned["activation"]
print(X.shape, y.shape)
```

```
(1373560, 67) (1373560,)
```

In [ ]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
```

Random Forest

In [ ]:

```
from cuml.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=100,
                             max_depth=16)
```

In [ ]:

```
rfc.fit(X_train, y_train)
```

Out[ ]:

```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [ ]:

```
from cuml.metrics import accuracy_score

y_train_pred = rfc.predict(X_train)
rf_train_accuracy = accuracy_score(y_train, y_train_pred)
print("Train accuracy: ", rf_train_accuracy)

y_test_pred = rfc.predict(X_test)
rf_test_accuracy = accuracy_score(y_test, y_test_pred)
print("Test accuracy: ", rf_test_accuracy)
```

```
Train accuracy:  0.9867188334465027
Test accuracy:  0.9827929139137268
```

In [ ]:

```
from cuml.metrics import confusion_matrix

print("Train: ",confusion_matrix(y_train, y_train_pred, convert_dtype=Tru
print("Test: ",confusion_matrix(y_test, y_test_pred, convert_dtype=True))
```

```
Train:  [[1077149       0]
 [  14594    7105]]
Test:  [[269254    191]
 [  4536    731]]
```

In [ ]:

```
print("F1: ", calc_f1(confusion_matrix(y_test, y_test_pred, convert_dtype
```

```
F1:  0.991289301229659
```

In [ ]:

```
dump(rfc, 'rfc.joblib')
```

Out[ ]:

```
['rfc.joblib']
```

In [ ]:

SVM

In [ ]:

```
from cuml.svm import SVC

svm = SVC(kernel='rbf', class_weight='balanced')
```

In [ ]:

```
svm.fit(X_train, y_train)
```

Out[ ]:

```
▾ SVC

SVC()
```

In [ ]:

```
from cuml.metrics import accuracy_score

y_train_pred = svm.predict(X_train)
svm_train_accuracy = accuracy_score(y_train, y_train_pred)
print("Train accuracy: ", svm_train_accuracy)

y_test_pred = svm.predict(X_test)
svm_test_accuracy = accuracy_score(y_test, y_test_pred)
print("Test accuracy: ", svm_test_accuracy)
```

```
Train accuracy:  0.7885931730270386
Test accuracy:  0.7888770699501038
```

In [ ]:

```
from cuml.metrics import confusion_matrix

print("Train: ",confusion_matrix(y_train, y_train_pred, convert_dtype=Tru
print("Test: ",confusion_matrix(y_test, y_test_pred, convert_dtype=True))
```

```
Train:  [[853296 223853]
 [  8451  13248]]
Test:  [[213519  55926]
 [  2072   3195]]
```

In [ ]:

```
print("F1: ", calc_f1(confusion_matrix(y_test, y_test_pred, convert_dtype
```

```
F1:  0.8804253704879638
```

In [ ]:

```
dump(svm, 'svm.joblib')
```

Out[ ]:

```
['svm.joblib']
```